REMARKS:

## Status

Claims 1 to 53, 80 to 110, and 124 to 164 are pending. No claims have been amended, added, or deleted. Claims 1, 18, 36, 80, 90, 100, 110, 124, 125, 126, 134, 142, 152, 153, and 157 are the independent claims. Reconsideration and further examination are respectfully requested.

## Section 103 Rejection

Claims 1 to 53, 80 to 110, and 124 to 164 were rejected under 35 USC § 103 over U.S. Patent No. 6,151,618 (Wahbe) in view of U.S. Patent Pub. No. 2001/0037450 (Metlitski). Applicant respectfully traverses this rejection. The claims are discussed below grouped by independent claims.

Claims 1 to 17: Independent claim 1 is reproduced below:

> 1. A method of analyzing instructions and data for a program to determine where the instructions and data might result in incorrect results when run on a multiprocessor system, the method comprising the steps of:
>
> dividing the instructions and data for the program into plural domains based on symbols used to refer to those instructions and data, the multiprocessor system configured to use at most one processor at a time to execute instructions and to access data from any one domain;
>
> determining which of the instructions and data involve references outside of their domains;
>
> determining which of the references outside of their domains are multiprocessor unsafe references;
>
> generating a report of the multiprocessor unsafe references; and
>
> modifying the instructions and data based on the report.

The applied art, alone or in combination, does not teach the foregoing features of claim 1, at least with respect to "dividing the instructions and data for the program into plural domains based on symbols used to refer to those instructions and data."

The Office Action acknowledged that "Wahbe doesn't explicitly disclose dividing the instructions and data for the program into plural domains based on symbols used to refer to those instructions and data." The Office Action cited Metlitski as remedying this deficiency. However, Metlitski does not teach this claim feature.

In more detail, the Office Action stated the following: "However, Metlitski in an analogous art and similar configuration teaches a protected protocol being split in an open architecture which includes two virtual machine architecture see [0113, item 150] and for splitting protected process see [0115 – 0116]." However, this is not what is claimed. Rather, the claim language states "dividing the instructions and data for the program into plural domains **based on symbols used to refer to those instructions and data.**"

Page 3 of the specification for the pending application emphasizes that "[t]his technique is different from known old techniques in which domains are defined based on locations of instructions and data within a particular piece of software." According to Applicant's reading, Metlitski's operation is akin to these old techniques.

Metlitski at paragraph [0116] teaches that "[t]he source code 4 of a software product is divided into segments S1 . . . Sn from which segments desired to be protected 41 (e.g., those critical to program execution) are selected. Next, the segments to be executed using the trusted module are compiled using the CryptoCompiler 8." Earlier in paragraph [0114], Metlitski states that "[i]n the exemplary embodiment of the protected application development, a set of program segments is identified. Of course, one skilled in the art will recognize that the selection of these segments is dependent on a particular application as a matter of design choice." Applicant understands these segments to be the same segments referred to in paragraph [0116]. Thus, the selection of the segments in Metlitski is not based on symbols used to refer to instructions and data, as claimed, but rather is "a matter of design choice."

The difference between the claimed technique and the teachings of Metlitski is emphasized by the fact that the word "symbol" does not even appear in Metlitski.

In view of the foregoing, reconsideration and withdrawal are respectfully requested of the § 103 rejection of claim 1 and its dependent claims over the applied art. Allowance of these claims also is respectfully requested.

Claims 18 to 35: Independent claim 18 is reproduced below:

> 18. A memory storing information including steps executable by a processor, the steps executable to analyze instructions and data for a program to determine where the instructions and data might result in incorrect results when run on a multiprocessor system, the steps comprising:
>
> dividing the instructions and data for the program into plural domains based on symbols used to refer to those instructions and data, the multiprocessor system configured to use at most one processor at a time to execute instructions and to access data from any one domain;
>
> determining which of the instructions and data involve references outside of their domains;
>
> determining which of the references outside of their domains are multiprocessor unsafe references;
>
> generating a report of the multiprocessor unsafe references; and
>
> modifying the instructions and data based on the report.

Substantially as discussed above with respect to claim 1, the applied art does not teach the foregoing features of claim 18, at least with respect to "dividing the instructions and data for the program into plural domains based on symbols used to refer to those instructions and data." Accordingly, allowance of claim 18 and its dependent claims over the applied art is respectfully requested.

<u>Claims 36 to 53</u>: Independent claim 36 is reproduced below:

> 36. An analyzer that analyzes instructions and data for a program to determine where the instructions and data might result in incorrect results when run on a multiprocessor system, the analyzer comprising:
>
> a reference analyzer that divides the instructions and data for the program into plural domains based on symbols used to refer to those instructions and data, the multiprocessor system configured to use at most one processor at a time to execute instructions and to access data from any one domain, that determines which of the instructions and data involve references outside of their domains, and that determines which of the references outside of their domains are multiprocessor unsafe references; and
>
> a report generator that generates a report of the multiprocessor unsafe references.

Substantially as discussed above with respect to claim 1, the applied art does not teach the foregoing features of claim 36, at least with respect to "a reference analyzer that divides the instructions and data for the program into plural domains based on symbols used to refer to those instructions and data." Accordingly, allowance of claim 36 and its dependent claims over the applied art is respectfully requested.

<u>Claims 80 to 89</u>: Independent claim 80 is reproduced below:

> 80. A method of analyzing instructions and data and dynamically determining where the instructions and data for a program might result in

incorrect results when run on a multiprocessor system, the method comprising the steps of:

dividing the instructions and data for the program into plural domains based on symbols used to refer to those instructions and data, the multiprocessor system configured to use at most one processor at a time to execute instructions and to access data from any one domain;

determining which of the instructions and data involve references outside of their domains;

determining which of the references outside of their domains are purportedly multiprocessor safe references;

generating a table of the purportedly multiprocessor safe references, the table including the domains to which the references are supposed to refer;

executing the instructions and data; and

when a reference in the table of purportedly microprocessor safe references is encountered during execution of the instructions and data, determining if the reference is actually to a domain to which that reference is supposed to refer.

Substantially as discussed above with respect to claim 1, the applied art does not teach the foregoing features of claim 80, at least with respect to "dividing the instructions and data for the program into plural domains based on symbols used to refer to those instructions and data." Accordingly, allowance of claim 80 and its dependent claims over the applied art is respectfully requested.

Claims 90 to 99: Independent claim 90 is reproduced below:

90. A memory storing information including steps executable by a processor, the steps executable to analyze instructions and data for a program and dynamically determine where the instructions and data might result in incorrect results when run on a multiprocessor system, the steps comprising:

dividing the instructions and data for the program into plural domains based on symbols used to refer to those instructions and data, the multiprocessor system configured to use at most one processor at a time to execute instructions and to access data from any one domain;

determining which of the instructions and data involve references outside of their domains;

determining which of the references outside of their domains are purportedly multiprocessor safe references;

generating a table of the purportedly multiprocessor safe references, the table including the domains to which the references are supposed to refer;

executing the instructions and data; and

when a reference in the table of purportedly microprocessor safe references is encountered during execution of the instructions and data, determining if the reference is actually to a domain to which that reference is supposed to refer.

Substantially as discussed above with respect to claim 1, the applied art does not teach the foregoing features of claim 90, at least with respect to "dividing the instructions and data for the program into plural domains based on symbols used to refer to those instructions and data." Accordingly, allowance of claim 90 and its dependent claims over the applied art is respectfully requested.

Claims 100 to 109: Independent claim 100 is reproduced below:

100. A system for analyzing instructions and data for a program and dynamically determining where the instructions and data might result in incorrect results when run on a multiprocessor system, the system comprising:

a reference analyzer that divides the instructions and data for the program into plural domains based on symbols used to refer to those instructions and data, the multiprocessor system configured to use at most one processor at a time to execute instructions and to access data from any one domain, that determines which of the instructions and data involve references outside of their domains, and that determines which of the references outside of their domains are purportedly multiprocessor safe references;

a table generator that generates a table of the purportedly multiprocessor safe references, the table including the domains to which the references are supposed to refer;

> a reference tracker that tracks references made by the instructions and data; and
>
> a comparator that determines, when a reference in the table of purportedly microprocessor safe references is encountered during execution of the instructions and data, if the reference is actually to a domain to which that reference is supposed to refer.

Substantially as discussed above with respect to claim 1, the applied art does not teach the foregoing features of claim 100, at least with respect to "a reference analyzer that divides the instructions and data for the program into plural domains based on symbols used to refer to those instructions and data." Accordingly, allowance of claim 100 and its dependent claims over the applied art is respectfully requested.

Claim 110: Independent claim 110 is reproduced below:

> 110. A method of analyzing instructions and data for a program to determine where the instructions and data might result in incorrect results when run on a system having multiple resources of a type used concurrently, the method comprising the steps of:
>
> dividing the instructions and data for the program into plural domains based on symbols used to refer to those instructions and data, the system configured to use at most one of the resources at a time to execute instructions and to access data from any one domain;
>
> determining which of the instructions and data involve references outside of their domains;
>
> determining which of the references outside of their domains are unsafe references;
>
> generating a report of the unsafe references; and
>
> modifying the instructions and data based on the report.

Substantially as discussed above with respect to claim 1, the applied art does not teach the foregoing features of claim 110, at least with respect to "dividing the instructions and data for the program into plural domains based on symbols used to refer to those instructions and data." Accordingly, allowance of claim 110 over the applied art is respectfully requested.

Claim 124: Independent claim 124 is reproduced below:

> 124. A report stored in a memory, the report resulting from analysis of instructions and data for a program to determine where the instructions and data might result in incorrect results when run on a multiprocessor system, the report comprising:
>
> a division of the instructions and data for the program into plural domains based on the symbols used to refer to those instructions and data, the multiprocessor system configured to use at most one processor at a time to execute instructions and to access data from any one domain;
>
> a list of inter-domain references by the instructions and data that the analysis has shown are multiprocessor unsafe; and
>
> for each inter-domain reference, the domains involved in the inter-domain reference.

Substantially as discussed above with respect to claim 1, the applied art does not teach the foregoing features of claim 124, at least with respect to "a division of the instructions and data for the program into plural domains based on the symbols used to refer to those instructions and data." Accordingly, allowance of claim 124 over the applied art is respectfully requested.

Claim 125: Independent claim 125 is reproduced below:

> 125. A table stored in a memory, the table resulting from analysis of instructions and data for a program to determine where the instructions and data might result in incorrect results when run on a multiprocessor system, the report comprising:
>
> a division of the instructions and data for the program into plural domains based on the symbols used to refer to those instructions and data, the multiprocessor system configured to use at most one processor at a time to execute instructions and to access data from any one domain;
>
> a list of purportedly microprocessor safe references by the instructions and data outside of their domains; and
>
> the domains to which the references are supposed to refer.

Substantially as discussed above with respect to claim 1, the applied art does not teach the foregoing features of claim 125, at least with respect to "a division of the instructions and data for the program into plural domains based on the symbols used to refer to those instructions and data." Accordingly, allowance of claim 125 over the applied art is respectfully requested.

Claims 126 to 133: Independent claim 126 is reproduced below:

> 126. A method of dynamically analyzing instructions and data for a program to determine where the instructions and data result in domain violations when run on a multiprocessor system, the method comprising the steps of:
>
> dividing the instructions and data for the program into plural domains based on symbols used to refer to those instructions and data, the multiprocessor system configured to use at most one processor at a time to execute instructions and to access data from any one domain;
>
> accessing a table of purportedly microprocessor safe references by the instructions and data outside of their domains, the table including the domains to which the references are supposed to refer;
>
> executing the instructions and data; and
>
> when a reference in the table of purportedly microprocessor safe references is encountered during execution of the instructions and data, determining if the reference is actually to a domain to which that reference is supposed to refer.

Substantially as discussed above with respect to claim 1, the applied art does not teach the foregoing features of claim 126, at least with respect to "dividing the instructions and data for the program into plural domains based on symbols used to refer to those instructions and data." Accordingly, allowance of claim 126 and its dependent claims over the applied art is respectfully requested.

Claims 134 to 141: Independent claim 134 is reproduced below:

> 134. A memory storing information including steps executable by a processor, the steps executable to dynamically analyze instructions and data for a program to determine where the instructions and data result in domain violations when run on a multiprocessor system, the steps comprising:

dividing the instructions and data for the program into plural domains based on symbols used to refer to those instructions and data, the multiprocessor system configured to use at most one processor at a time to execute instructions and to access data from any one domain;

accessing a table of purportedly microprocessor safe references by the instructions and data outside of their domains, the table including the domains to which the references are supposed to refer;

executing the instructions and data; and

when a reference in the table of purportedly microprocessor safe references is encountered during execution of the instructions and data, determining if the reference is actually to a domain to which that reference is supposed to refer.

Substantially as discussed above with respect to claim 1, the applied art does not teach the foregoing features of claim 134, at least with respect to "dividing the instructions and data for the program into plural domains based on symbols used to refer to those instructions and data." Accordingly, allowance of claim 134 and its dependent claims over the applied art is respectfully requested.

Claims 142 to 151: Independent claim 142 is reproduced below:

142. A checker that dynamically analyzes instructions and data for a program to determine where the instructions and data result in domain violations when run on a multiprocessor system, the checker comprising:

an analyzer that divides the instructions and data for the program into plural domains based on symbols used to refer to those instructions and data, the multiprocessor system configured to use at most one processor at a time to execute instructions and to access data from any one domain;

an interface to a table of purportedly microprocessor safe references by the instructions and data outside of their domains, the table including the domains to which the references are supposed to refer; and

a reference tracker that tracks references made by the instructions and data; and

a comparator that determines, when a reference in the table of purportedly microprocessor safe references is encountered during execution of the instructions and data, if the reference is actually to a domain to which that reference is supposed to refer.

Substantially as discussed above with respect to claim 1, the applied art does not teach the foregoing features of claim 142, at least with respect to "an analyzer that divides the instructions and data for the program into plural domains based on symbols used to refer to those instructions and data." Accordingly, allowance of claim 142 and its dependent claims over the applied art is respectfully requested.

Claim 152: Independent claim 152 is reproduced below:

152. A method of dynamically analyzing instructions and data for a program to determine where the instructions and data result in domain violations when run on a system having multiple resources of a type used concurrently, the method comprising the steps of:
dividing the instructions and data for the program into plural domains based on symbols used to refer to those instructions and data, the system configured to use at most one of the resources at a time to execute instructions and to access data from any one domain;
accessing a table of purportedly safe references by the instructions and data outside of their domains, the table including the domains to which the references are supposed to refer;
executing the instructions and data; and
when a reference in the table of purportedly safe references is encountered during execution of the instructions and data, determining if the reference is actually to a domain to which that reference is supposed to refer.

Substantially as discussed above with respect to claim 1, the applied art does not teach the foregoing features of claim 152, at least with respect to "dividing the instructions and data for the program into plural domains based on symbols used to refer to those instructions and data." Accordingly, allowance of claim 152 over the applied art is respectfully requested.

Claims 153 to 156: Independent claim 153 is reproduced below:

> 153. Instructions and data for a program stored in a memory, the instructions and data analyzed and configured for execution on a multiprocessor system, comprising:
> domain definitions whereby the instructions and data for the program are defined as existing in plural domains; and
> annotations whereby references outside of their domains in the instructions and data are indicated as being multiprocessor safe.

The applied art, alone or in combination, does not teach the foregoing features of claim 153, at least with respect to "annotations whereby references outside of their domains in the instructions and data are indicated as being multiprocessor safe."

In this regard, the Office Action cited Fig. 4, 422 and 418, of Wahbe as follows: "for reference outside of their domains see '*identify foreign program.*'" Wahbe's step 418 in Fig. 4 is "identify foreign program for execution or transfer." Wahbe's step 422 determines if the "foreign program is safe." Wahbe, col. 7, lines 57 to 58. However, this use of "safe" is not equivalent to the claimed "multiprocessor safe." In fact, Wahbe does not even mention multiprocessor operation.

Furthermore, in the elaboration of step 422 discussed with respect to Fig. 5, Wahbe does not teach using annotations. Wahbe instead mentions annotations in the context of optimization, as follows:

> Local and global optimization techniques are commonly used by compilers to improve the program flow and/or performance due to specific instruction combinations, but were heretofore not used in virtual machine implementations because they require more hardware architecture specific information at compile time than a virtual machine can provide. The presence and/or incorporation of annotations results in a performance optimization, however, annotations are not a requirement for the implementation of program optimization or safety. The effect of annotations on a computer program is to give the virtual machine implementation more information

> about the nature of an untrusted program such as what parts of the program are intended to be executable and what parts are intended to be read-only data. The annotation information does not have to b trusted for safety implementations to work however. A virtual machine implementation can use annotation information to load and run a computer program more efficiently.

Wahbe, col. 12, lines 33 to 51. This text refers to "performance optimization," not to indicating that references are "multiprocessor safe" as claimed.

Thus, Wahbe does not teach claim 153's feature of "annotations whereby references outside of their domains in the instructions and data are indicated as being multiprocessor safe."

Metlitski makes no mention of annotations and therefore does not remedy the foregoing deficiency of Wahbe.

In view of the foregoing, reconsideration and withdrawal are respectfully requested of the § 103 rejection of claim 153 and its dependent claims over the applied art. Allowance of these claims also is respectfully requested.

Claims 157 to 164: Independent claim 157 is reproduced below:

> 157. Annotations in instructions and data for a program stored in a memory, the instructions and data analyzed and configured for execution on a multiprocessor system, the instructions and data for the program divided into plural domains based on symbols used to refer to those instructions and data, the multiprocessor system configured to use at most one processor at a time to execute instructions and to access data from any one domain, the annotations comprising reasons why inter-domain references in the instructions and data for the program are multiprocessor safe, whereby analysis of the instructions and data considers the annotated references to be multiprocessor safe.

The applied art, alone or in combination, does not teach the foregoing features of claim 157. Substantially as discussed above with respect to claim 1, the applied art does not teach the feature that "the instructions and data for the program divided into plural domains based on symbols used to refer to those instructions and data." Similarly as discussed above with respect to claim 153, the applied art does not teach the feature that "annotations comprising reasons why inter-domain references in the instructions and data for the program are multiprocessor safe."

In view of the foregoing, reconsideration and withdrawal are respectfully requested of the § 103 rejection of claim 157 and its dependent claims over the applied art. Allowance of these claims also is respectfully requested.

## No Admission

Applicant's decision not to argue each of the dependent claims separately is not an admission that the subject matter of those claims is taught by the applied art.

## Closing

In view of the foregoing amendments and remarks, the entire application is believed to be in condition for allowance, and such action is respectfully requested at the Examiner's earliest convenience.

Applicant's undersigned attorney can be reached at (614) 205-3241. All correspondence should continue to be directed to the address indicated below.

Respectfully submitted,

Dane C. Butzer

Dated: December 19, 2005

Dane C. Butzer
Reg. No. 43,521

Swernofsky Law Group PC
P.O. Box 390013
Mountain View, CA 94039-0013
(650) 947-0700